Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Информационные системы»

УТВЕРЖДАЮ

Заведующий кафедрой ИС
_____ С. А. Виденин
«__» _____ 2016 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Разработка ИС для центра дендрохронологических исследований ИКИТ
(frontend-разработка портала)

Руководитель _____ С.А. Виденин

Выпускник _____ А.В. Костюк

Нормоконтролер _____ Ю. В. Шмагрис

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В середине 2015 года проект научного коллектива Сибирского федерального университета «Цифровая микроанатомия годичных колец в физиологии, экологии, климатологии и археологии» (далее — заказчик) выиграл грант Российского научного фонда по итогам конкурса, направленного на поддержку проектов и осуществление фундаментальных и поисковых научных исследований, результаты которых учёные представят на международных конференциях.

В результате учёные рассчитывают разработать новые и улучшить имеющиеся методы количественной оценки климатических и экологических условий, физиологического отклика деревьев в прошлом по данным цифровой микроанатомии годичных колец.

В рамках этого гранта можно выделить ряд проблем:

- информация, содержащаяся в структуре годичных колец, на сегодняшний день почти не используется в мире;
- методы анализа полученных в исследованиях данных на данный момент не достаточно эффективны и довольно медлительны;
- получение измерений микроанатомии годичных колец деревьев в настоящее время — недостаточно автоматизированный процесс;
- отсутствие единой структуры хранения измерений;
- проблема использования чужих актуальных измерений в своих исследованиях.

Следовательно, объектом исследования является микроанатомия годичных колец, а предметом исследования является эффективность методов хранения и обработки ее данных.

В рамках этого гранта необходимо разработать информационную систему (далее — ИС) и базу данных (далее — БД), чтобы загружать, хранить и обрабатывать данные, полученные в ходе исследований. Поэтому целью данной

выпускной квалификационной работы является повышение эффективности методов хранения и обработки данных микроанатомии годичных колец.

Для достижения поставленной цели решаются следующие задачи:

- выявление и анализ требований к ИС;
- проектирование ИС;
- проектирование и разработка БД цифровой микроанатомии годичных колец;
- разработка ИС.

1 Обзор предметной области

Дендрохронология — научное направление, которое возникло на стыке таких наук, как биология, экология, география, климатология, археология, лесоводство и других.

Эту науку часто подразделяют на дендроклиматологию, дендроиндикацию и собственно дендрохронологию.

Дендрохронология — изучение изменчивости ширины годичных колец деревьев для точного датирования времени их образования [1, с. 5].

Дендроиндикация — выявление естественных колебаний природных процессов и экологически значимых антропогенных изменений на основе реакции на них древесных растений и их сообществ.

Дендроклиматология — изучение годичных колец деревьев с целью оценки изменения климата.

Дендрохронологический анализ применяется во многих областях науки:

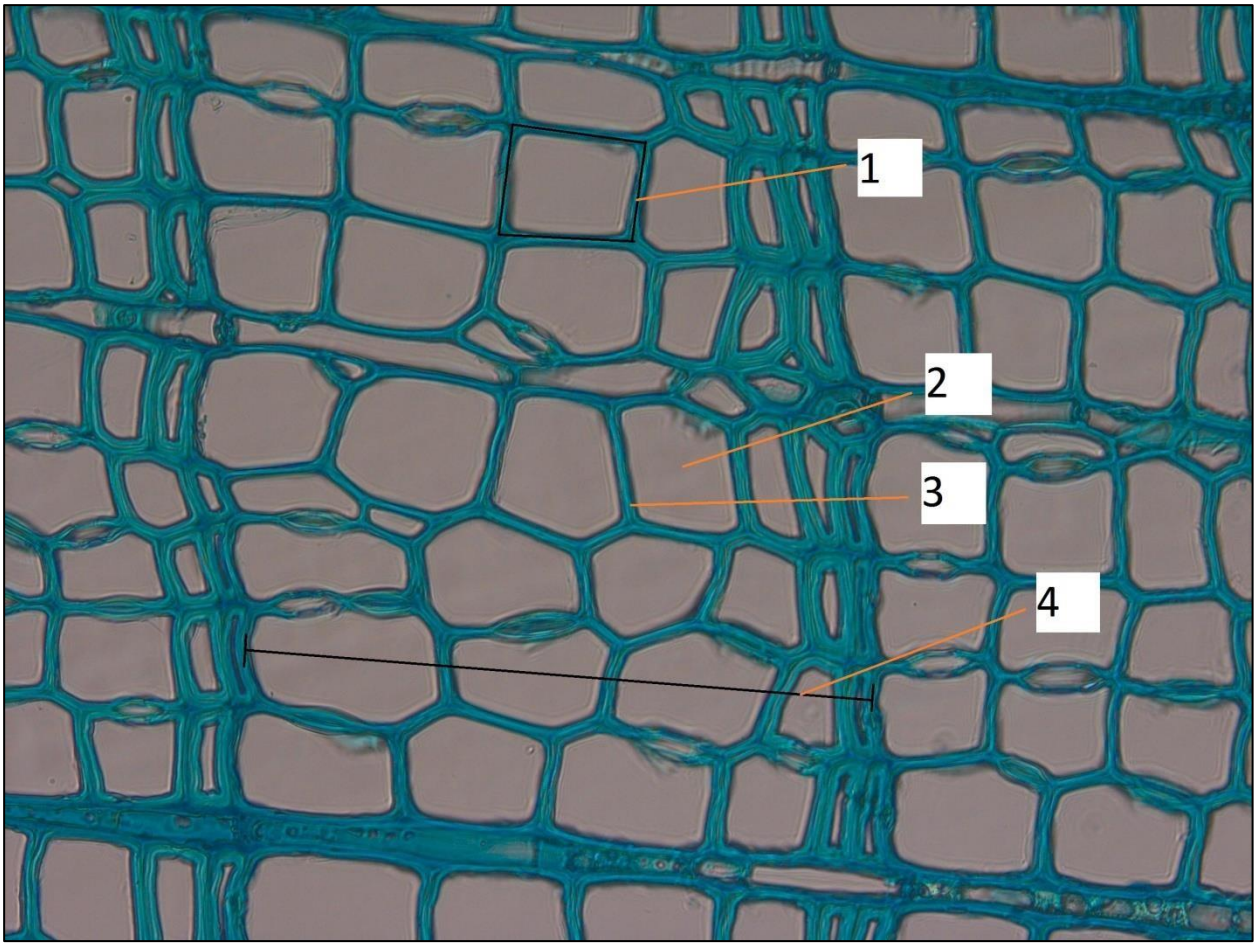
- в геофизике и астрономии (установление солнечно-земных связей);
- в палеоэкологии и археологии (установление климата и экологических условий в прошлом);
- в климатологии и метеорологии (установление климатических условий и сверхдолгосрочное прогнозирование);
- для изучения влияния климатических условий на изменение природы леса и появление лесных пожаров;
- для изучения возрастной структуры насаждений, смены пород, естественного возобновления;
- для прогнозирования продуктивности лесов и оценки лесохозяйственных мероприятий;
- для оценки влияния климата на водные ресурсы, речной сток и уровень воды в водоемах;
- для индикации частоты схода лавин и селей;
- для датирования вулканических извержений;

- в судебно-биологической экспертизе (контроль за законностью вырубки древесины).

Не смотря на то, что дендрохронология как наука появилась в середине двадцатого века, потенциал годичных колец в качестве косвенного источника экологической и климатической информации до сих пор еще не исчерпан.

Годичные кольца деревьев обладают внутренней структурой, содержащей дополнительную экологическую информацию, которую можно извлечь, и которая практически не используется в мире на данный момент.

На рисунке 1 представлено изображение структуры годичных колец дерева.



1 — клетка; 2 — люмен; 3 — клеточная стенка; 4 — ширина годичного кольца

Рисунок 1 — Внутренняя структура годичных колец дерева

Анализ количественных показателей микроанатомической структуры годичных колец показателей позволит не только повысить точность и разрешающую способность климатических реконструкций, но и оценить функционально-физиологический отклик деревьев на изменчивость климатических условий, а, значит, поможет понять реакцию древесной растительности на прогнозируемые климатические изменения и их влияния на социум.

1.1 Дендрохронологические исследования в Сибирском федеральном университете

Продолжением исследований в области цифровой микроанатомии годичных колец деревьев с приложениями для физиологии, климатологии и экологии, а также расширением этих исследований занимаются ученые Сибирского федерального университета в рамках гранта «Цифровая микроанатомия годичных колец в физиологии, экологии, климатологии и археологии» (номер — 15-14-30011) под руководством доктора исторических наук Мыглана Владимира Станиславовича [2].

Ожидаемы результаты работы по этому гранту:

- Разработка новых и улучшение имеющихся методов для количественной оценки климатических и экологических условий и физиологического отклика деревьев в прошлом по количественным и качественным показателям микроанатомии годичных колец.

- Установление преимуществ и недостатков методов оценки климата прошлого по цифровым микроанатомическим данным о структуре годичных колец. Описание характера изменений климата за последние два с половиной тысячелетия вдоль трансекта Алтай-Тыва на основе результатов, полученных при реконструкции динамики климатических параметров и экстремальных событий в окружающей среде по данным ширины и микроанатомии структуры годичных колец.

- Разработка математических моделей для оценки функциональных характеристик (проведения, метаболических затрат, механической прочности) клеточной структуры годичных хвойных. Построение временных рядов изменчивости функциональных характеристик годичных колец деревьев лиственницы вдоль трансекта Алтай-Тыва в Центральной Азии.

- Разработка архитектуры и открытой для пользования БД по цифровой микроанатомии годичных колец.

- Описание закономерности влияния внешних условий на изменчивость микроанатомических и функционально-физиологических характеристик годичных колец.

Результаты работы в дальнейшем могут использоваться для продолжения исследований в понимании механизмов и явлений, лежащих в основе современных климатических изменений, оценки вариаций изменений в темпах роста, распространения и выживаемости древесной растительности при прогнозируемых климатических изменениях, а также их технологические и социальные последствия.

Также результаты работы могут быть применены для решения ряда проблем фундаментального характера:

- сверхдлинные хронологии ширины годичных колец различных деревьев могут быть использованы для датировки археологической и исторической древесины;

- построение прогнозов в сельском хозяйстве и страховом бизнесе, где требуется учитывать риски экстремальных климатических проявлений;

- построение прогнозов при проектировании и обслуживании технических сооружений, представляющих повышенную опасность для окружающей среды.

Кроме того, все результаты могут использоваться в сфере среднего и высшего профессионального образования, например, в научно-исследовательских работах студентов, аспирантов и молодых ученых, или в улучшении программ обучения магистров и аспирантов по специальностям

экология, физиология растений, климатология, археология, этнография, история, рациональное использование природных ресурсов, а также во многих других областях.

Подводя итоги всего вышесказанного можно выделить ряд проблем:

- микроанатомия годичных колец деревьев несет в себе более подробную экологическую и климатическую информацию, нежели просто измерения ширины годичных колец, но эта информация на сегодняшний день почти не используется в мире;

- методы количественной оценки климатических и экологических условий и физиологического отклика деревьев в прошлом по полученным данным на данный момент не достаточно эффективны и довольно медлительны;

- получение измерений микроанатомии годичных колец деревьев в настоящее время — недостаточно автоматизированный процесс, не лишенный погрешностей, а также требующий постоянного участия человека в процессе измерений и, следовательно, большого количества времени на измерение;

- также отсутствие единого банка данных затрудняет использование чужих актуальных измерений в своих исследованиях, останавливая всю исследовательскую работу на несколько месяцев;

- отсутствие единого банка данных влечет за собой отсутствие единой структуры хранения измерений, что негативно влияет на единообразность представления данных, простоту их описания и обработки.

Один из примеров нынешнего оформления результатов представлен на рисунке 2.

Как можно заметить, структура записей значительно отличается, что в дальнейшем затрудняет процесс автоматизации обработки хранимых данных. Чтобы решить эту проблему, необходимо разработать единую структуру записи, которой могли бы придерживаться все исследователи этой области.

	A	B	C	D	E	F	G	H
1	исходные измерения отдельных рядов и средняя трахеидограмма							
2	год	ширина годич	№ клетки в радиальном фа	толщина клеточной стенки, мкм				
3			номер клетк	PPT	TKC	PPT	TKC	PPT
4	тангентальный размер ряда						30	
5				средняя трахеидограмм		ряд 1		ряд 2
6	2003	790,744	1	43,38	2,77	43,38	2,77	43,38
7	2003	790,744	2	46,59	2,42	46,59	2,42	46,59
8	2003	790,744	3	44,05	2,58	44,05	2,58	44,05
9	2003	790,744	4	43,60	2,88	43,60	2,88	43,60
10	2003	790,744	5	41,71	3,34	41,71	3,34	41,71
11	2003	790,744	6	37,61	3,78	37,61	3,78	37,61
12	2003	790,744	7	36,16	3,77	36,16	3,77	36,16
13	2003	790,744	8	37,36	3,68	37,36	3,68	37,36
14	2003	790,744	9	32,84	3,90	32,84	3,90	32,84
15	2003	790,744	10	28,12	5,36	28,12	5,36	28,12
16	2003	790,744	11	27,27	7,29	27,27	7,29	27,27
17	2003	790,744	12	27,02	8,26	27,02	8,26	27,02
18	2003	790,744	13	29,54	9,08	29,54	9,08	29,54

Рисунок 2 — Пример оформления результатов измерений

Еще один пример оформления для сравнения представлен на рисунке 3.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	pin 1					pin 4					pin 7			
2	year	cell	D	CWT		year	cell	D	CWT		year	cell	D	CWT
3	1969	1	37,16	2,035		1969	1	38,69	1,91		1972	1	44,0	2,165
4	1969	2	49,11	2,29		1969	2	42,00	2,29		1972	2	50,9	2,55
5	1969	3	59,04	2,925		1969	3	40,47	2,29		1972	3	55,1	2,93
6	1969	4	45,81	2,8		1969	4	39,70	2,545		1972	4	35,4	3,19
7	1969	5	44,53	3,435		1969	5	30,80	2,545		1972	5	41,8	2,43
8	1969	6	43,77	2,545		1969	6	27,23	3,055		1972	6	52,4	3,45
9	1969	7	41,48	2,545		1969	7	34,62	2,545		1972	7	53,2	2,80
10	1969	8	36,14	2,545		1969	8	34,39	2,925		1972	8	49,1	2,42
11	1969	9	30,28	2,8		1969	9	30,84	2,545		1972	9	48,9	2,42
12	1969	10	25,45	3,565		1969	10	32,83	2,8		1972	10	55,5	2,55
13	1969	11	18,07	3,945		1969	11	35,63	3,055		1972	11	38,4	2,67
14	1969	12	14,50	3,435	**0,445**	1969	12	34,11	3,18		1972	12	39,2	2,80
15						1969	13	36,66	3,055		1972	13	42,0	2,55
16	1970	1	37,69	2,42		1969	14	38,45	3,18		1972	14	45,6	3,18
17	1970	2	33,39	2,43		1969	15	33,67	3,565		1972	15	45,4	3,44
18	1970	3	36,94	1,91		1969	16	23,67	4,835		1972	16	46,3	3,31
19	1970	4	34,95	2,67		1969	17	18,08	4,33		1972	17	49,1	3,57
20	1970	5	35,64	2,29		1969	18	10,69	3,565		1972	18	40,7	3,69
21	1970	6	32,83	3,31		1969	19	11,45	2,925	0,594	1972	19	29,3	5,09
22	1970	7	30,79	2,035							1972	20	22,4	5,86
23	1970	8	19,60	2,165		1970	1	30,80	3,565		1972	21	17,6	5,09
24	1970	9	22,39	2,67		1970	2	32,57	2,8		1972	22	16,5	3,95
25	1970	10	41,98	2,67		1970	3	35,12	2,42		1972	23	7,4	2,93
26	1970	11	31,81	2,035		1970	4	36,40	2,67					
27	1970	12	26,47	3,31		1970	5	35,88	2,67		1973	1	41,5	3,055
28	1970	13	13,49	3,31	**0,398**	1970	6	31,31	2,42		1973	2	47,3	2,67
29						1970	7	28,51	3,055		1973	3	47,9	2,8
30	1971	1	33,35	1,915		1970	8	34,36	2,925		1973	4	44,1	2,545
31	1971	2	29,77	2,67		1970	9	35,64	2,8		1973	5	48,4	2,545

Рисунок 3 — Пример оформления результатов измерений

Данные проблемы уже пытались решить, однако лишь в части задач удалось добиться успеха. Из существующих решений, больше всех задач удалось выполнить международному банку данных колец деревьев — сервису The International Tree-Ring Data Bank (далее — ITRDB).

ITRDB — сервис, созданный в 1990 году палеоклиматологической программой Национального управления океанических и атмосферных исследований США. Благодаря этому сервису были сохранены в одном месте все доступные наборы данных, которых в настоящий момент около 6 000 штук. Эти данные были собраны из более чем 1 500 сайтов по всему миру [3].

В сервисе ITRDB любой исследователь может скачать и загрузить данные. Сервис практически не регламентирует какие метаданные пользователь может вводить при загрузке данных и в каком виде, что в последствии затрудняет поиск для других людей. Также сервис не регламентирует в каком виде должны быть загружены данные измерений, что затрудняет впоследствии их изучение другими исследователями в своих работах.

На рисунке 4 представлен интерфейс поиска данных в этом сервисе. Можно производить поиск по автору, что загрузил данные, по стране, отмечать желаемые параметры измерений и, как основной параметр, выбирать вид дерева.

Плюсом является поддержка множественного выбора — можно производить поиск по нескольким авторам публикация, видам деревьев одновременно. В разрабатываемом сервисе также будет поддержка множественного выбора.

Интерфейс сложный для неопытного пользователя, а также недостаточно эффективен.

Как итог — данный сервис не решает большую часть поставленных задач и представляет из себя просто FTP-сервер, не предоставляющий средств для обработки загруженной информации и не регламентирующий, в каком виде должны быть загружены данные.

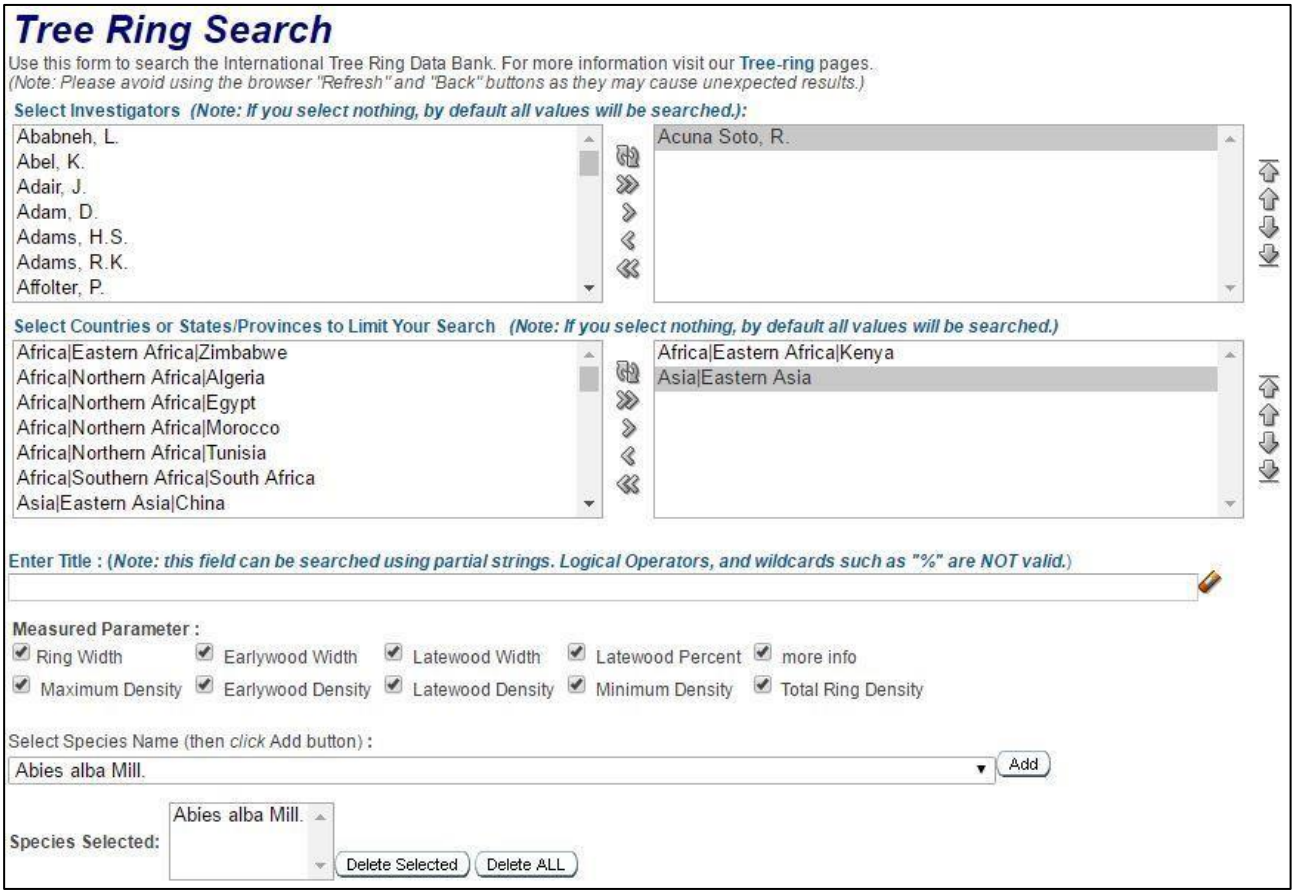


Рисунок 4 — Интерфейс поиска в сервисе ITRDB

В дополнении к этим параметрам, присутствуют еще несколько, представленных на рисунке 5. Здесь можно вводить годы, которые интересуют исследователя, а также примерные координаты месторасположения дерева.

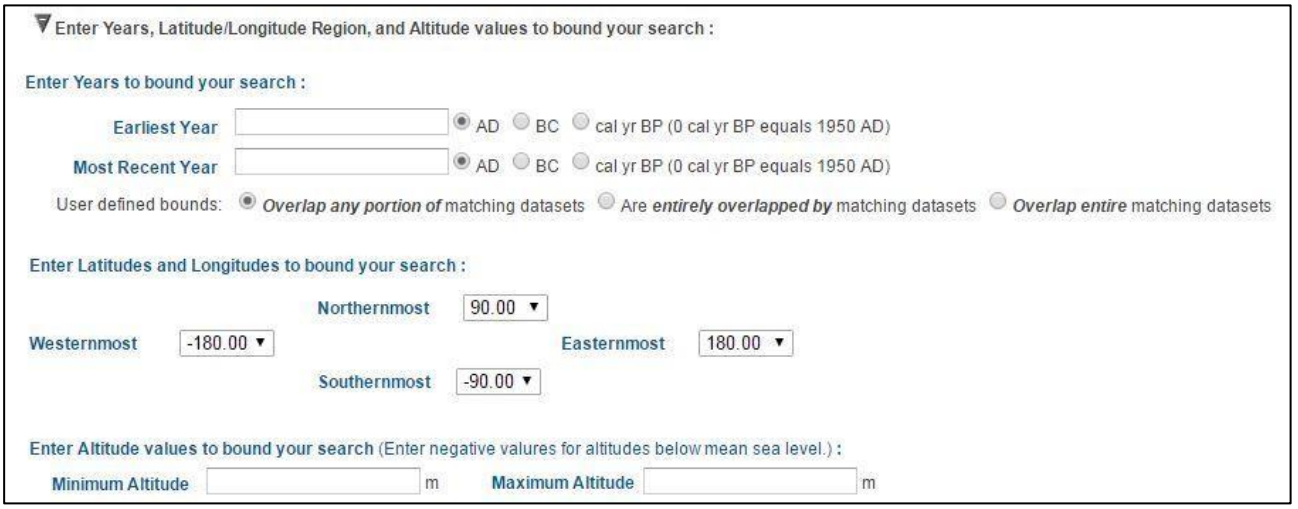


Рисунок 5 — Дополнительные параметры поиска в сервисе ITRDB

В таблице 1 представлено сравнение между тем, что могут предложить существующие решения и тем, что требуется достичь. Требуемые параметры более подробно расписаны в главе 2.

Таблица 1 — Сравнение существующих решений и требований

Критерий сравнения	Существующие решения	Требуемые параметры
Наличие стандартов	Нет единой стандартизации, что затрудняет ученым использовать эти данные.	Стандартизированный файл с измерениями.
Объем данных	Объёмы, доступные для хранения данных, зависят от возможностей каждого отдельного пользователя или ограниченны сервисом.	Сохранение большого количества снимков объёмом равному примерно двум гигабайтам.
Поиск данных	Поиск по тексту. Результат не гарантирован, т.к. наличие каких-либо комментариев и пояснений при сохранении данных не требуется. Минимальное количество атрибутов, к примеру вид дерева. Неудобный поиск месторасположения дерева по координатам.	Гарантированный успех поиска по атрибутам при наличии самих данных благодаря требованию ввода обязательных атрибутов перед публикацией. Предварительный просмотр превью-изображения снимка. Виджет интерактивной спутниковой карты для указания критерия поиска месторасположения дерева
Анализ снимков	В основном платные решения, которые, как и бесплатные, слабо автоматизированы и не могут дать нужного результата. Необходимость иметь высокую вычислительную мощность у машины, на которой будет проводиться анализ. Результаты анализа требуют последующей обработки перед использованием.	Анализ снимков проходит автоматически и соответствует всем требованиям. Результат не требует последующей обработки. Анализ проводится на сервере, поэтому от клиента требуется только предоставить снимок. Высокая скорость анализа.

Проанализируя таблицу можно сделать вывод, что существующие решения не удовлетворяют поставленным требованиям и нуждаются во множестве доработок.

1.2 Вывод по главе 1

В связи с вышеперечисленным можно выделить объект исследования — микроанатомию годичных колец, предмет исследования — эффективность методов хранения и обработки данных микроанатомии годичных колец.

Исходя из предмета исследования ставится цель данной выпускной квалификационной работы — повышение эффективности методов хранения и обработки данных микроанатомии годичных колец.

Чтобы достигнуть поставленной цели, необходимо выполнить ряд задач:

- выявление и анализ требований к ИС;
- проектирование ИС;
- проектирование и разработка базы данных цифровой микроанатомии годичных колец;
- разработка ИС.

2 Выявление и анализ требований к информационной системе

Чтобы выяснить, что из себя должна представлять ИС, необходимо провести интервью с заказчиком и установить функциональные и нефункциональные требования к ИС. В процессе встреч был сформирован список предварительных требований, который впоследствии будет пополняться новыми пунктами.

2.1 Функциональные требования

К функциональным требованиям разрабатываемой ИС относятся следующие требования:

- ИС должна производить авторизацию пользователя путем ввода идентификационного логина и пароля (после предварительной единовременной регистрации);
- ИС должна предоставлять определенные права пользователям, в зависимости от их статуса («пользователь», «администратор», «модератор», «новичок», «заблокирован» и незарегистрированный пользователь);
- ИС должна предоставлять пользователю «заблокирован» те же права, что и незарегистрированному:

1 возможность фильтрации данных по атрибутам;
2 просмотр списка загруженных данных;
3 просмотр «превью» изображения данных вместе с атрибутами;
4 просмотр и скачивание литературы;
5 просмотр профилей пользователей;
6 чтение лицензионного соглашения;

- ИС должна предоставлять пользователю «новичок», помимо тех же прав, что и незарегистрированному пользователю, следующие права:

1) загрузка новых данных измерений (файлы измерений, изображение, большие основные изображения, атрибуты);
2) редактирование загруженных данных;

3) редактирование своего профиля (контактная информация и т. п.);

4) возможность восстановления пароля;

5) возможность смены пароля;

- ИС должна предоставлять пользователю «пользователь», помимо тех же прав, что и пользователю «новичок», следующие права:

6) просмотр списков загруженных и скачанных пользователем данных;

7) скачивание уже загруженных пользователями данных;

8) загрузка и редактирование новой литературы, а также возможность ее удаления;

- ИС должна предоставлять пользователю «модератор», помимо тех же прав, что и пользователю «пользователь», следующие права:

9) доступ в кабинет управление пользователями с возможностью просмотра списка пользователей;

10) возможность изменение статуса пользователя (кроме статуса «администратор») из той страны, за которую он назначен модератором;

11) возможность удаления и редактирования загруженных данных пользователями из той страны, за которую он назначен модератором;

12) возможность просмотра полной информации профиля пользователей (отображение даже скрытых от обычных пользователей номеров телефонов и адреса электронной почты);

13) возможность редактировать и удалять литературу, загруженную пользователями той страны, за которую он назначен модератором;

- ИС должна предоставлять пользователю «администратор», помимо тех же прав, что и пользователю «модератор», следующие права:

14) возможность изменения статуса любого пользователя на любой другой;

15) возможность редактирования и удаления любой литературы;

16) возможность редактирования и удаления любых данных, загруженных пользователями;

17) доступ в кабинет управления атрибутами (возможности добавления и удаления атрибутов, а также редактирования справочников атрибутов).

Функциональные требования можно передать наглядно с помощью диаграмм прецедентов (Use Case Diagram) [4].

Основное отличие этих диаграмм при различных ролях пользователей это ограничение в наборе доступных функций.

К примеру, на рисунке 5 для пользователей доступна функция «просмотра данных», но если ее расписать, то выяснится, что она имеет подфункцию «скачать данные», которая недоступна для пользователя, не прошедшего модерацию.

На рисунке 6 представлена диаграмма прецедентов для пользователя «заблокирован» или незарегистрированного.

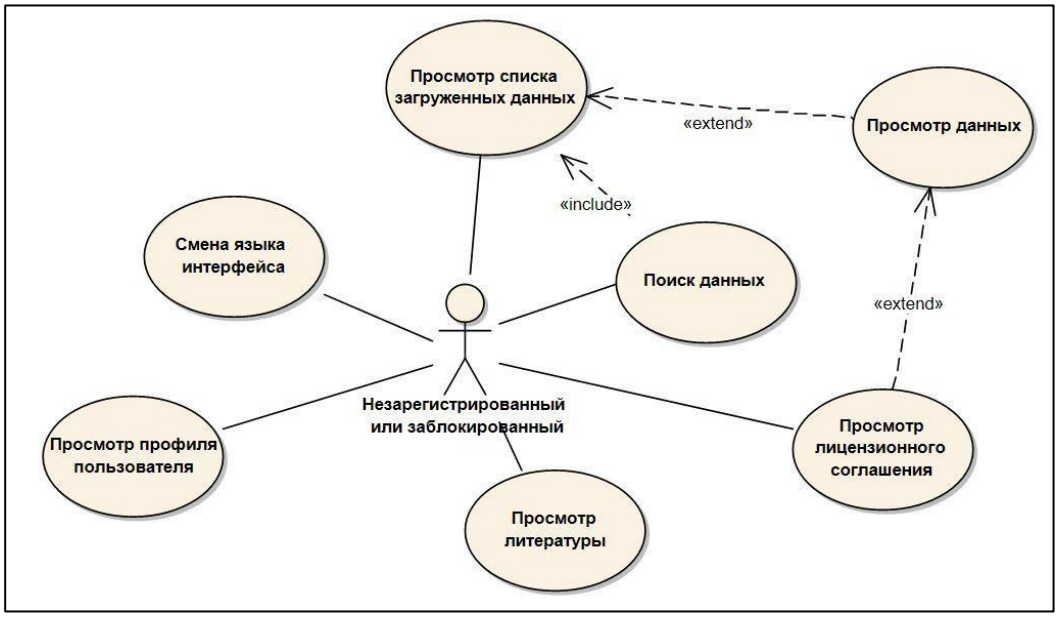


Рисунок 6 — Диаграмма прецедентов для пользователя «заблокирован» или незарегистрированного

В диаграмме прецедентов для пользователя «заблокирован» или незарегистрированного опущены такие действия как авторизация и регистрация, но сами функции в ИС присутствуют.

На рисунке 7 представлена диаграмма прецедентов для пользователя «новичок» или пользователя «пользователь».

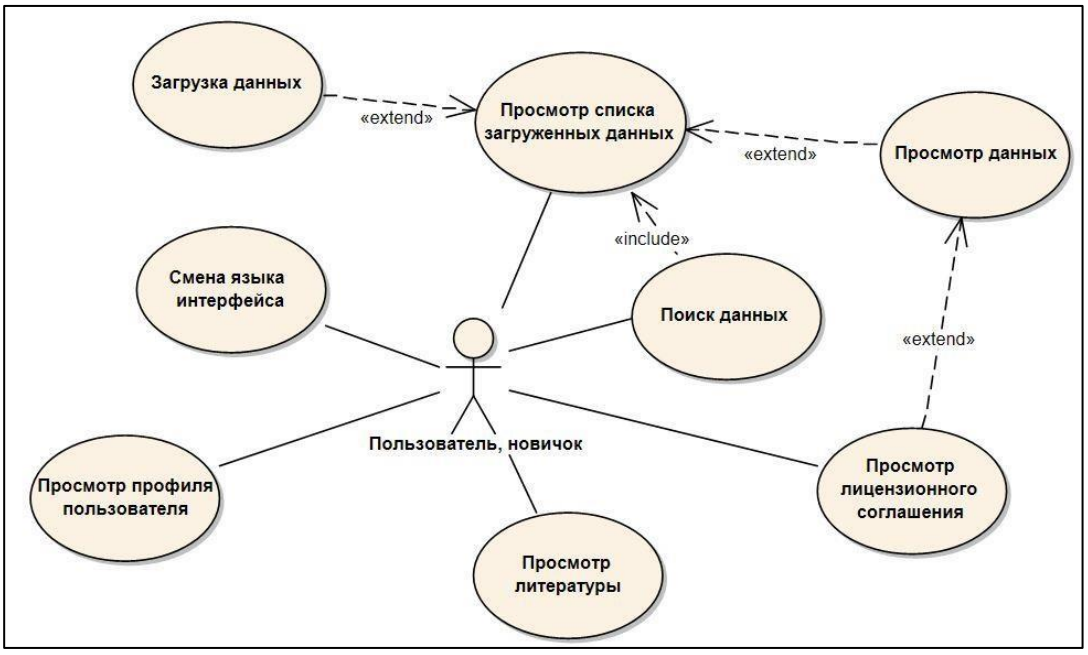


Рисунок 7 — Диаграмма прецедентов для пользователя «пользователь», прошедшего или не прошедшего модерацию

На рисунке 8 представлена диаграмма прецедентов для пользователя «модератор».

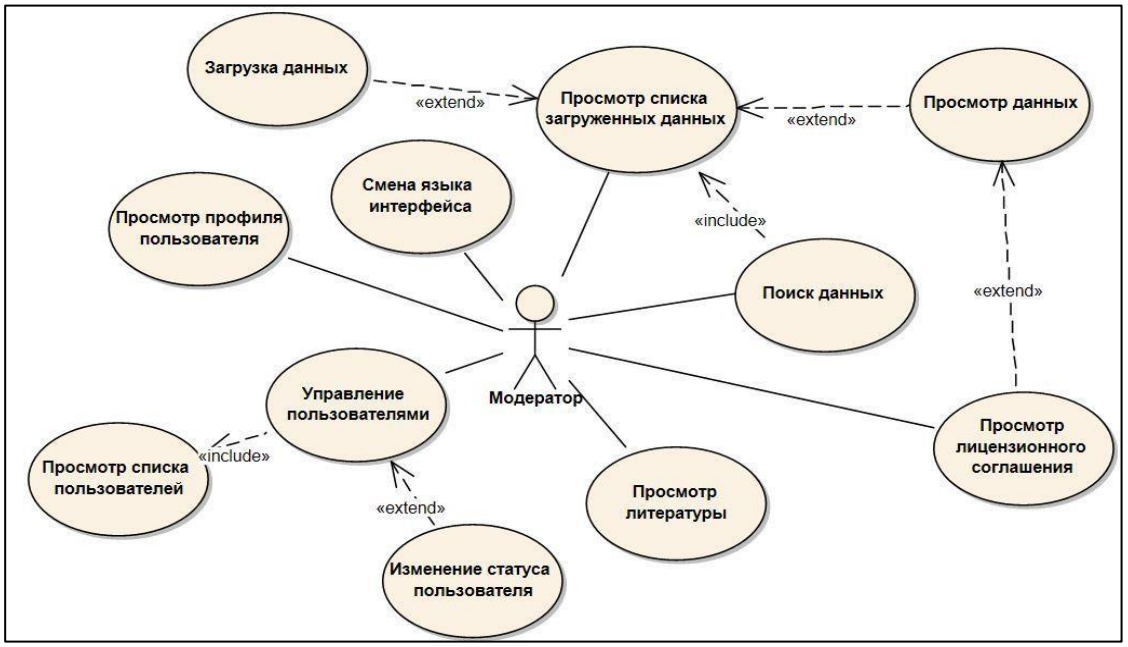


Рисунок 8 — Диаграмма прецедентов для пользователя «модератор»

На рисунке 9 представлена диаграмма прецедентов для пользователя «администратор».

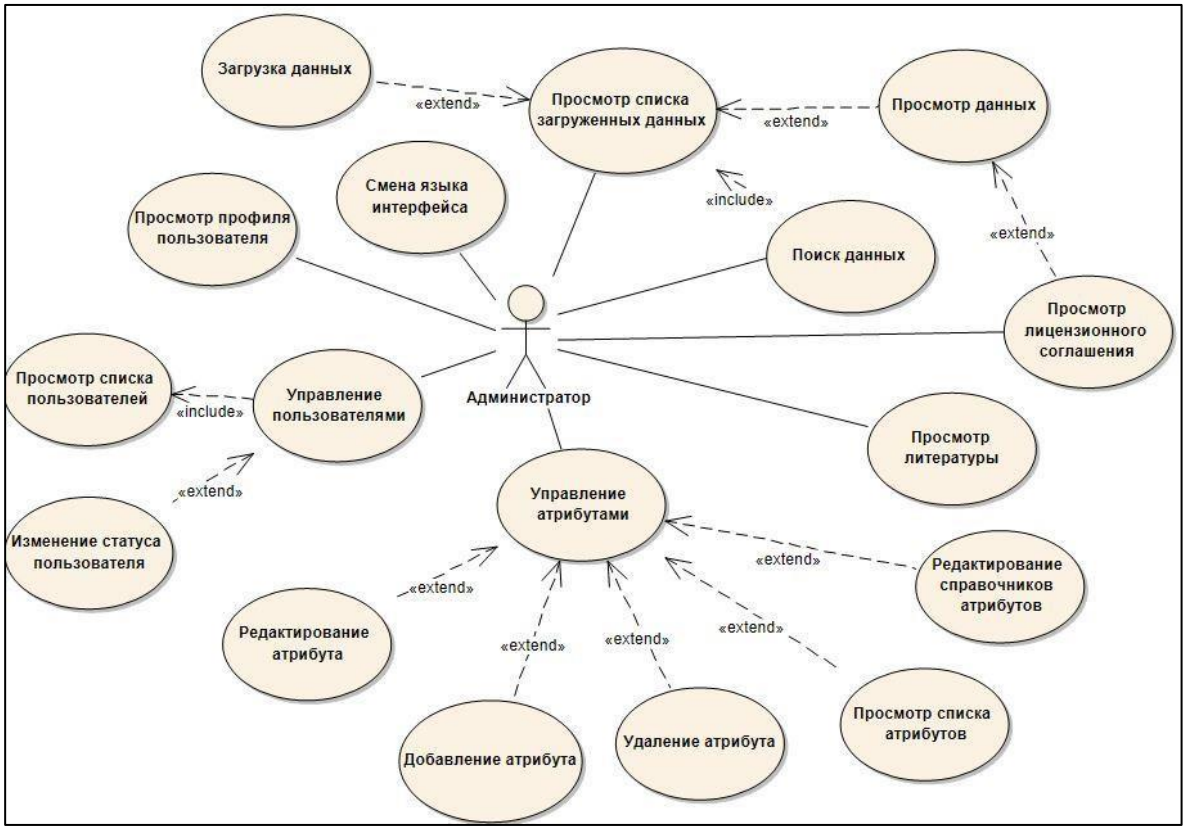


Рисунок 9 — Диаграмма прецедентов для пользователя «администратор»

На основе диаграмм прецедентов можно построить диаграмму классов для демонстрации классов системы, их атрибутов, методов и взаимосвязей между ними, чтобы в дальнейшем, при проектировании БД системы и разработке системы, опираться на нее. Построенная диаграмма классов представлена на рисунке 10.

Необходимо понимать, что данная диаграмма классов не содержит в себе все существующие классы системы, а только самые основные, чтобы показать, как устроена и должна работать система.

Например, класс *measurements* будет пополняться новыми атрибутами при дальнейшей разработке, а также будут добавляться новые классы для каждого из новых атрибутов.

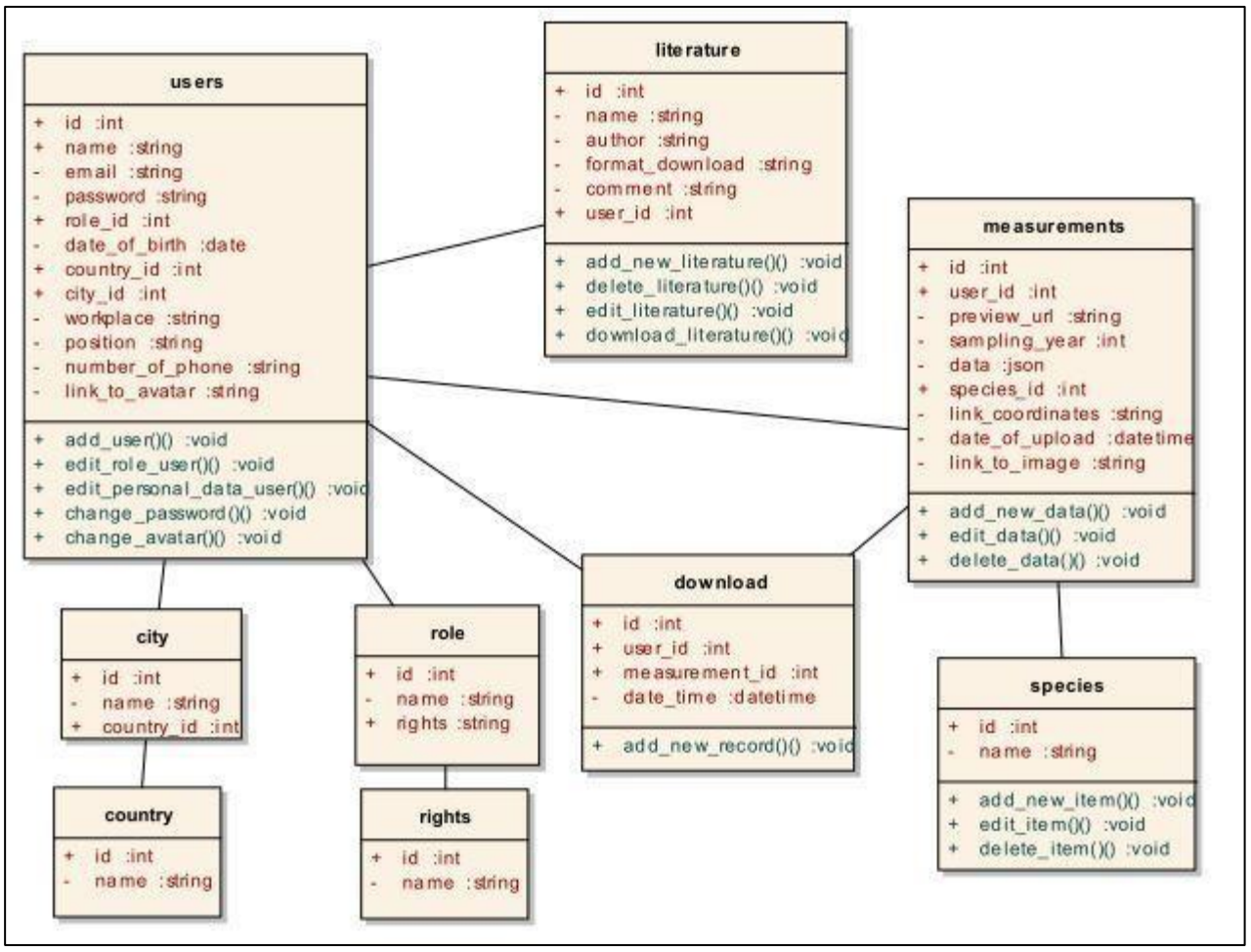


Рисунок 10 — Диаграмма классов

Чтобы показать подробно одну из возможных функций («скачивание данных»), изображена диаграмма последовательности на рисунке 11.

Алгоритм данной функции следующий:

- Авторизованный пользователь нажимает на кнопку просмотра загруженных данных;

- ИС обрабатывает запрос и отображает пользователю список данных;

- Пользователь выбирает интересующие его данные;

- ИС проверяет его права и, если ему не запрещено, отображает данные более подробно;

- Пользователь подтверждает принятие лицензионного соглашения и получает возможность скачать данные, если это разрешено;

- Происходит процесс передачи данных.

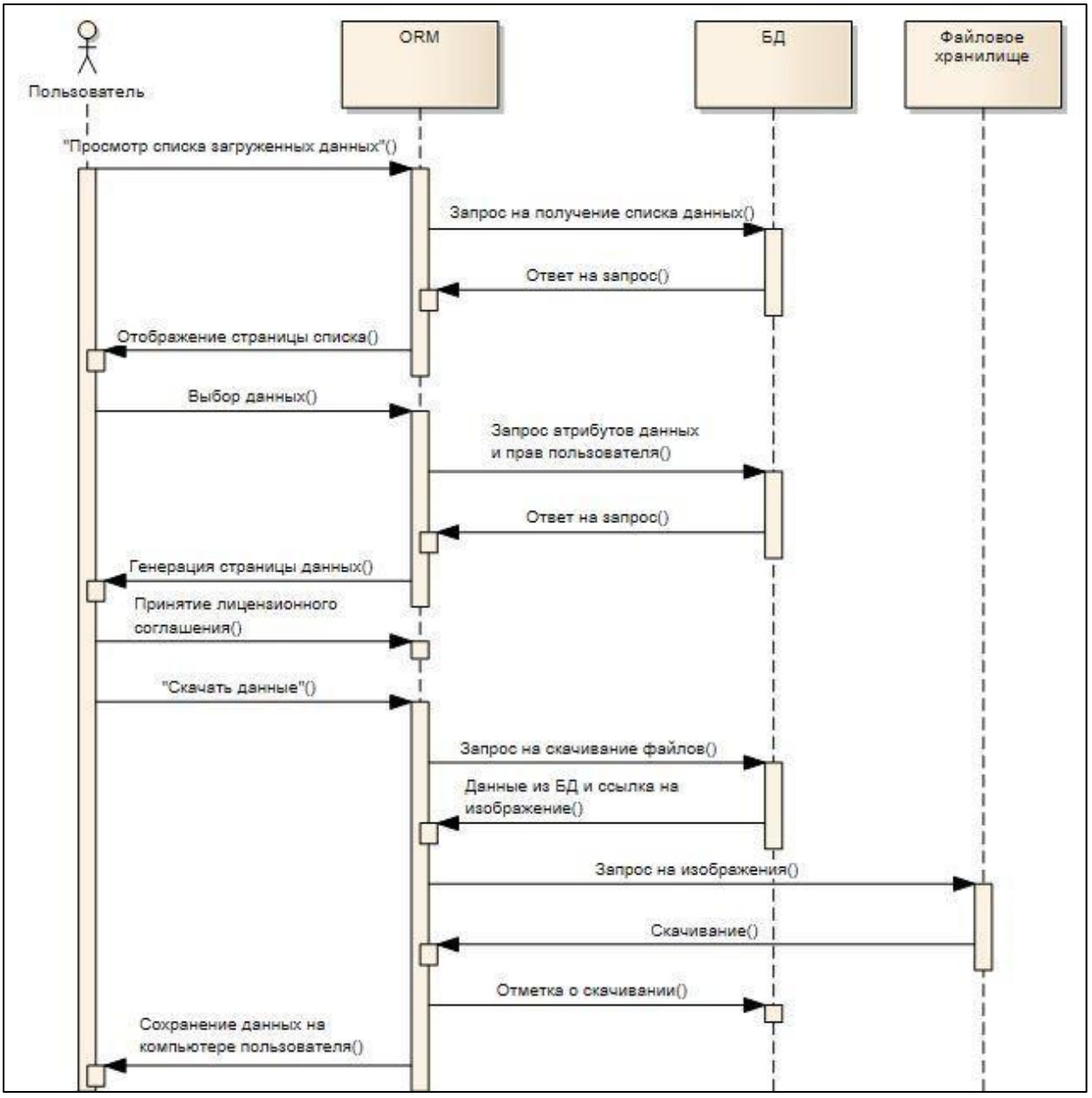


Рисунок 11 — Диаграмма последовательности функции «скачивание данных»

2.2 Нефункциональные требования

К нефункциональным требованиям разрабатываемой ИС относятся следующие требования:

- Требования к производительности:

1) ИС должна достаточно быстро обрабатывать информацию, так как объем обрабатываемых данных огромен — сотни тысяч строк измерений в файлах измерений и огромный размер изображений (до нескольких гигабайт);

2) ИС должна справляться с такими нагрузками, как работа с несколькими пользователями одновременно;

- Требования к безопасности:

3) Так как работать с ИС будет множество человек, то у каждого пользователя должен быть индивидуальный логин (адрес его электронной почты) и пароль (выбранный им самим);

4) ИС должна быть централизованной, то есть все данные должны находиться в едином хранилище;

5) ИС должна вести отслеживание всех изменений для контроля пользователей и работоспособности всей системы;

- Эксплуатационные требования:

6) ИС должна быть кроссплатформенна — должна работать в операционных системах семейств Linux и Windows;

7) Возможность доступа к ИС в любой точке мира, где есть выход в интернет;

- Политические и юридические требования:

8) Данные будут доступны по лицензии Creative Commons CC BY или CC BY SA NC, что значит, что их можно свободно скачивать, использовать и т.п., но с обязательным указанием авторства и ссылки на источник;

9) ИС должна быть реализована с помощью свободного программного обеспечения;

- Требования к пользовательскому интерфейсу:

10) ИС должна иметь такой интерфейс, чтобы все операции были интуитивно понятны людям, впервые пользующихся данной ИС;

11) ИС должна поддерживать два языка пользовательского интерфейса — русский и английский;

12) Интерфейс ИС должен быть выполнен в корпоративном стиле СФУ;

13) Интерфейс должен соответствовать функциональным требованиям.

2.3 Вывод по главе 2

Итогом этой главы являются сформированные функциональные и нефункциональные требования к ИС, а также построенные диаграммы прецедентов. Помимо этого, была построена диаграмма классов, которая впоследствии используется в разработке ИС и БД системы. Чтобы показать алгоритм работы одной из функций системы, в главе отображена диаграмма последовательности.

3 Разработка информационной системы

В данной главе рассматривается разработка ИС. Описывается проектирование ИС и БД системы, а также подробно разбираются основные модули ИС.

3.1 Проектирование информационной системы

Так как в требованиях к ИС существует возможность одновременного доступа к ИС нескольких человек, находящихся в любой точке мира, то система будет представлять из себя сервис в сети Интернет.

ИС разделяются на файл-серверные и клиент-серверные. В файл-серверных ИС на файловом сервере находится только БД системы, а сами приложения и СУБД находятся на рабочих станциях пользователей. В клиент-серверных ИС на сервере располагаются БД системы и СУБД, а на рабочей станции пользователя только клиентские приложения. В случае использования клиент-серверной ИС повышается защита данных, так как на сервере можно организовать контроль полномочий, чтобы давать пользователям доступ к данным только в соответствии с их правами. Также снижаются требования к рабочим станциям пользователя так как все вычисления производятся на сервере. Безусловно выбор падает на клиент-серверную архитектуру.

Для эффективного использования возможностей серверов и клиентов разумно будет использовать архитектуру Model-view-controller (MVC, модель-представление-контроллер) [5]. В данном случает ИС будет состоять из трех компонентов: клиента, сервера приложений (к нему подключено клиентское приложение) и слоя данных (с которым и будет работать сервер приложений). Поэтому модификация одного из компонентов будет оказывать минимальное воздействие на другие.

Архитектура ИС представлена на рисунке 12.

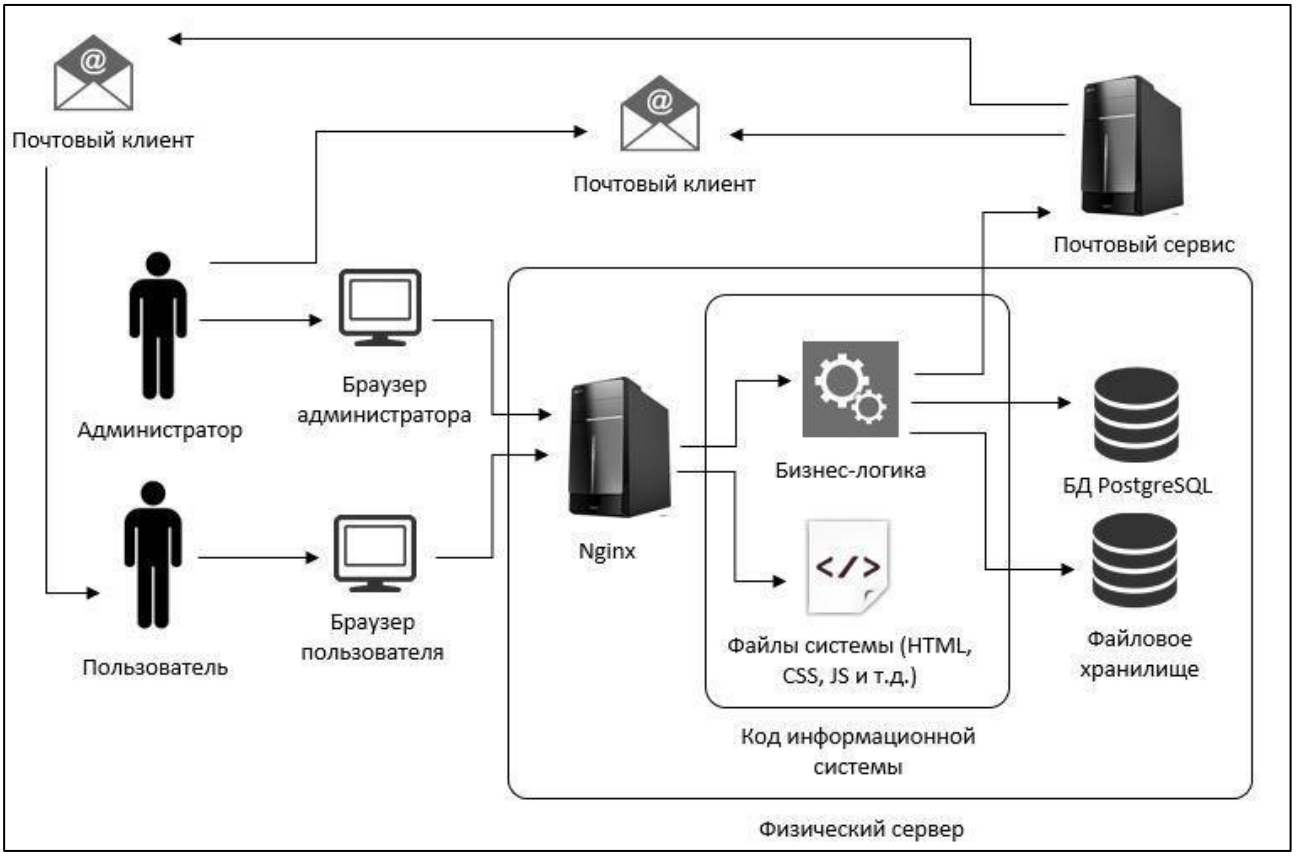


Рисунок 12 — Архитектура информационной системы

Проанализируя эту архитектуру, можно понять, из чего состоит разрабатываемая система, а также с чем и или кем она взаимодействует и посредством чего.

Например, пользователь решит восстановить пароль.

Пользователь зайдет через свой браузер на сервис и введет данные для восстановления пароля. Сервер примет эти данные, бизнес-логика их обработает и, посредством почтового сервиса, отправит пользователю данные для восстановления на его электронную почту. Пользователь, посредством своего браузера, использует эти данные, сервер примет их, а бизнес-логика проверит их корректность.

Кроме того, в ИС будет внедрена другая система, которая предназначена для распознавания изображений.

Модель интеграции этих информационных систем представлена на рисунке 13.

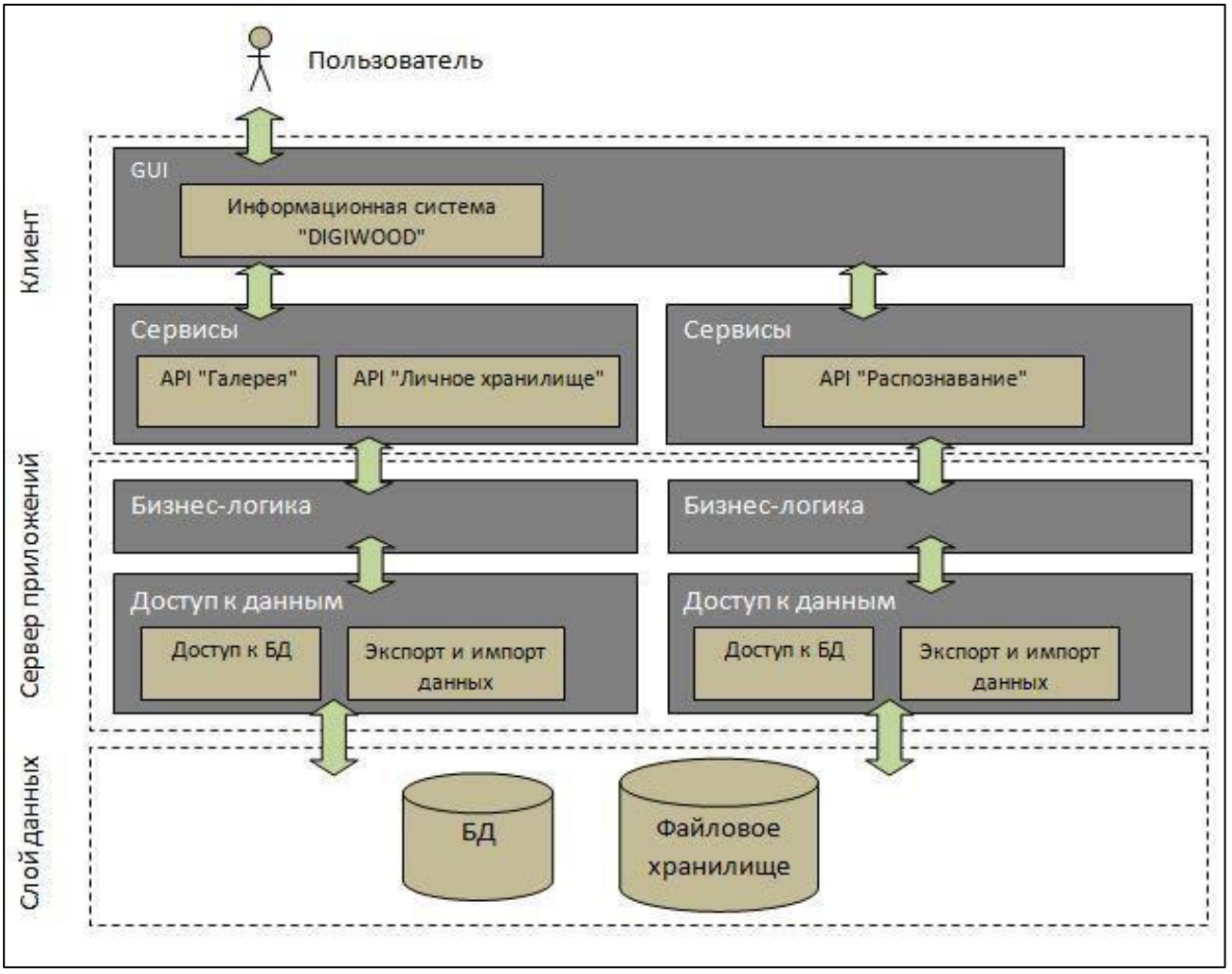


Рисунок 13 — Модель интеграции информационных систем

3.1.1 Слой клиента

Слой клиента будет представлять собой тонкий клиент — компьютер с браузером, который переносит все или большую часть задач по обработке информации на сервер.

Для реализации этого слоя будут использоваться HTML5 — язык разметки документов, CSS (каскадные таблицы стилей) — формальный язык описания внешнего вида документа, JavaScript — сценарный язык программирования, для обработки событий.

В этот слой входят все модули разрабатываемой системы, к которым имеет доступ пользователь, а также модуль, являющийся интегрированной ИС по распознаванию изображений.

3.1.2 Сервер приложений

В этот слой входит вся логика ИС. На этом уровне происходит обработка действий пользователя и осуществляется его связь с БД системы и файловым хранилищем.

Сервер приложений будет находиться на виртуальной машине Java (Java Virtual Machine, далее — JVM). JVM исполняет код Java. Веб-сервером является Nginx.

Языками программирования являются Java [6] и Scala [7]. Программы на языке Scala очень похожи на Java-программы и могут легко взаимодействовать с Java-кодом. По сути, язык Scala является приемником языка Java, технологично совершеннее, с меньшим объемом кода, с более развитым функциональным программированием.

В качестве среды разработки используется IntelliJ IDEA, которая ориентирована на написание программного обеспечения на языке Java.

В качестве каркаса всей системы, для избавления от создания рутинного кода и облегчения разработки, используется Play Framework — фреймворк, ориентированный на архитектуру MVC и созданный для языков Java и Scala [8].

Для построения web-сервиса и для описания HTTP-маршрутов в качестве языка DSL используется Spray [9].

Для генерации HTML-кода на стороне сервера используются библиотеки ScalaTags — библиотеки, позволяющие создавать шаблоны страниц [10].

Для связи сервера приложений с БД системы необходимо внедрить в него технологию объектно-реляционного отображения (object-relational mapping, далее — ORM). Ее задачей является преобразование объектов данных таким образом, чтобы существовала возможность сохранить их в БД системы, а также впоследствии извлечь без изменения свойств и отношений между ними. По сути, она позволяет абстрагироваться от способа хранения объектов и оперировать на уровне модели простыми объектами. Технология ORM избавляет программиста от написания большого количества кода, часто

однообразного и подверженного ошибкам, тем самым значительно повышая скорость разработки.

Существует множество видов технологий ORM программного обеспечения, которые можно разбить на группы, каждая из которых подходит для выбранного языка программирования (C++, Java, PHP, Perl, Ruby, Delphi и других). В свою очередь в каждой группе может существовать до нескольких десятков вариаций технологий ORM.

Так как в требованиях указано использование только свободного программного обеспечения, то выбор останавливается на языке Java, т.к. на изучение других, незнакомых языков программирования, будет потрачено достаточно много времени.

Список ORM-библиотек на языке Java довольно обширен (несколько десятков разновидностей) и на описание особенностей каждого из них ушло бы несколько десятков страниц, поэтому разумней будет указать выбранную ORM-библиотеку и ее возможности.

Ebean — это ORM-фреймворк с открытым исходным кодом, созданный на основе классических реализаций JPA (Java Persistence API) [11].

Из ключевых особенностей:

- поддержка типов PostgreSQL;
- простой для понимания API;
- простота в настройке;
- гибкая выборка связанных сущностей;
- частичные выборки;
- отслеживание изменений;
- собственная поддержка транзакций;
- асинхронная загрузка.

В целом ORM-фреймворк Ebean является простым в использовании и понимании, но в то же самое время увеличивает производительность за счет того, что совершает выборку только того, что пользователь указал в своем большом и сложном запросе.

3.1.3 Слой данных

Слой данных представляет собой саму БД системы и файловое хранилище. Далее происходит выявление и анализ требований к БД системы.

Требования к БД системы:

- Поддержка OLAP технологий;
- Поддержка форматов JSON, JSON_B;
- Кроссплатформенность;
- Возможности репликации и транзакции;
- Подзапросы;
- Возможность использования триггеров;
- Поддержка языка Java.

Базы данных по типу хранимой информации можно разделить на две категории: фактографические БД и документальные БД.

В фактографических БД содержатся короткие сведения об объектах, записанные в строго определенном формате.

В документальных БД информация может быть графической, звуковой, текстовой, мультимедийной, а единицей хранения является какой-либо документ.

Так как в требованиях к БД для хранения данных микроанатомии годичных колец имеется такое требование, как возможность хранения множества графических изображений большого размера, а также возможность хранения больших объемов текста, используемого в качестве описания характеристик объекта, то в данном случае необходимо использовать документальную БД.

Так как уже определено, что ИС будет являться клиент-серверной, то список возможных систем управления базами данных (далее — СУБД) будет следующим: Oracle, Firebird, Interbase, IBM DB2, Informix, MS SQL Server, Sybase Adaptive Server Enterprise, PostgreSQL, MongoDB, MySQL, Caché, ЛИНТЕР.

Далее рассмотрены отобранные СУБД с точки зрения требований к БД системы.

В целом, возможности всех рассматриваемых СУБД практически соответствуют заявленным требованиям, поэтому логично выделить лишь некоторые аспекты, на которые стоит обратить внимание.

Из перечисленных выше СУБД рассматриваются в дальнейшем те, которые возможно использовать без дополнительных трат денежных средств на получение лицензии — свободное программное обеспечение: MySQL, PostgreSQL, MongoDB, Firebird.

СУБД MySQL исключается по причине того, что она является реляционной, а нам требуется документальная, так как основные данные, хранимые в БД системы не будет (или практически не будет) связана между собой, а будут представлять из себя массив текстовых данных.

СУБД Firebird не устраивает по причине того, что по умолчанию в ней идет поддержка языка Delphi, а не Java.

В СУБД MongoDB отсутствуют транзакции.

В требованиях к БД системы есть пункт — поддержка форматов JSON, JSON_B. Это одно из самых важных требований к разрабатываемой БД системы, так как основные данные — измерения микроанатомии годичных колец деревьев, предполагается хранить именно в таком формате. Это связано с тем, что данные с измерениями, помести их в таблицу, могут занимать сотни тысяч строк, что негативно скажется на скорости обращения к ним, а также их обработке. В формате JSON же эти данные будут представлять из себя не что иное, как просто текст с разделителями, что положительно скажется на времени обработки запросов.

СУБД PostgreSQL поддерживает данный формат. Поэтому в качестве СУБД для данной ИС используется PostgreSQL [12].

Файловое хранилище будет представлять из себя компьютер с несколькими SSD и HDD, которые необходимы для хранения большого объема информации, загружаемой пользователями, и для хранения резервных копий.

3.2 Проектирование базы данных микроанатомии годичных колец

В базе данных должны храниться пользователи (*users*), страны (*country*), города (*city*), роли пользователей (*role*), права пользователей (*rights*), данные о скачивании (*download*), литература (*literature*), измерения и атрибуты (measurements), а также единообразные справочники по некоторым атрибутам, в частности вид дерева (*species*).

Для этого спроектирована структура БД, изображенная на рисунке 14.

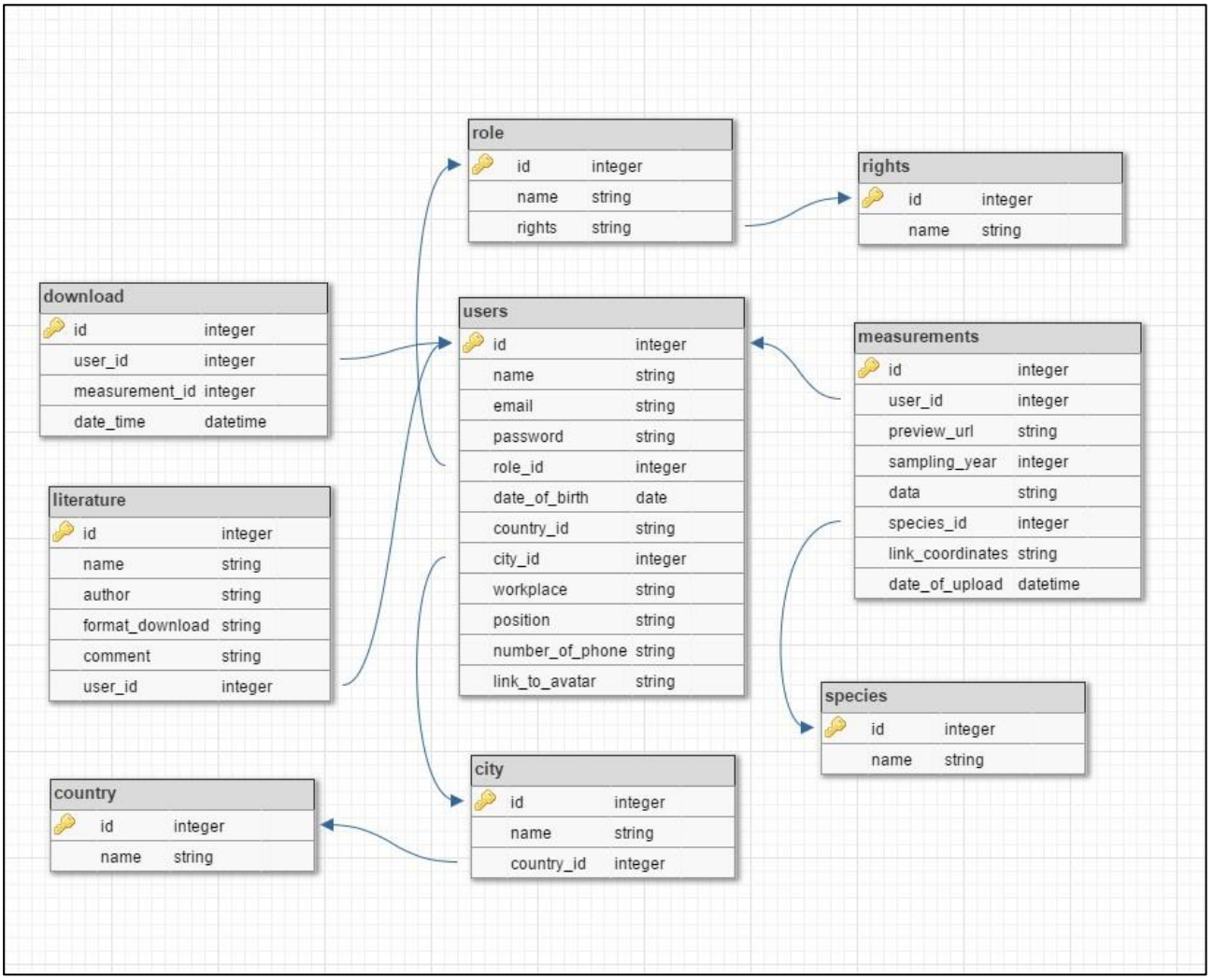


Рисунок 14 — Структура базы данных разрабатываемой системы

Каждая таблица имеет целочисленное поле id, которое является уникальным идентификатором записи и первичным ключом таблицы.

Далее представлены поля таблиц БД системы.

Таблица *users* содержит поля:

- Поле *name* — фамилия и имя;
- Поле *date_of_birth* — дата рождения;
- Поле *country_id* — *id* страны;
- Поле *city_id* — *id* города;
- Поле *workplace* — место работы/учебы;
- Поле *position* — должность;
- Поле *number_of_phone* — номер телефона;
- Поле *email* — адрес электронной почты (используется как логин);
- Поле *password* — пароль;
- Поле *role_id* — *id* роли;
- Поле *link_to_avatar* — ссылка на аватар пользователя.

Таблица *role* содержит поля:

- Поле *name* — название роли;
- Поле *rights* — массив *id* прав, представленный в виде строки.

Таблица *rights* содержит поле *name* — название права.

Таблица *country* содержит поле *name* — название страны.

Таблица *city* содержит поля:

- Поле *name* — название города;
- Поле *country_id* — *id* страны.

Таблица *download* содержит поля:

- Поле *measurement_id* — *id* измерения;
- Поле *user_id* — *id* пользователя;
- Поле *date_time* — время скачивания.

Таблица *literature* содержит поля:

- Поле *name* — название литературы;
- Поле *author* — автор литературы;
- Поле *format_download* — формат файла;
- Поле *comment* — комментарий;
- Поле *user_id* — *id* пользователя, который загрузил эту литературу.

Таблица *species* подержит поле *name* — название вида дерева.

Таблица *measurements* содержит поля:

- Поле *preview_url* — ссылка на превью-изображение;
- Поле *link_to_image* — ссылка на изображение;
- Поле *user_id* — *id* автора;
- Поле *species_id* — вид дерева;
- Поле *link_coordinates* — географические координаты;
- Поле *sampling_year* — год отбора образцов;
- Поле *date_of_upload* — дата загрузки;
- Поле *data* — данные измерений.

Представленная схема показывает, как будет храниться информация, обрабатываемая ИС.

3.3 Разработка графического пользовательского интерфейса

В данном разделе описывается разработка графического пользовательского интерфейса информационной системы, а также инструменты, которые при этом используются.

3.3.1 Выбор инструментов для разработки пользовательского интерфейса

В качестве среды разработки выбран PhpStorm, продукт компании JetBrains, разработанный на основе платформы IntelliJ IDEA. Данная среда разработки обеспечивает автодополнение, анализ кода на лету, навигацию по коду, отладку, рефакторинг и многое другое.

Для описания стилей используется каскадная таблица стилей — CSS.

Языком сценариев для интерактивности веб-страниц сервиса является JavaScript — прототипно-ориентированный сценарный язык программирования.

Для верстки страниц используются любые инструменты для создания веб-приложений и сайтов. Информация об инструментах заимствовалась с электронного ресурса htmlbook.ru [13].

3.3.2 Процесс разработки графического пользовательского интерфейса

С помощью интегрированной среды разработки PhpStorm был создан проект. Проект имеет следующую структуру:

- Директория *.idea*;
- Директория *Errors* — содержит страницы для ошибок 404 и т.п.;
- Директория *Images* — содержит изображения для страниц;
- Директория *Pages* — содержит страницы сайта;
- Директория *Scripts* — содержит скрипты для страниц;
- Директория *Style* — содержит CSS для стиля страниц;
- Файл *.htaccess*.

Директория *Pages* содержит в себе следующие страницы, по названию которых легко понять, что на них находится:

- Страница *Authorization*;
- Страница *Basement*;
- Страница *Create_new_data*;
- Страница *Data*;
- Страница *Edit_data*;
- Страница *Edit_password*;
- Страница *Edit_profile*;
- Страница *Gallery*;
- Страница *Header*;
- Страница *Landing_page*;
- Страница *License_agreement*;
- Страница *Literature*;
- Страница *Management*;

- Страница *Management_attributes*;

- Страница *Management_users*;

- Страница *Password_recovery*;

- Страница *Profile*;

- Страница *Registration*;

- Страница *Search*.

В процессе верстки был соблюден корпоративный стиль СФУ — преобладают оттенки серого и оранжевого цвета, а также присутствует символика университета.

На рисунке 15 представлено изображение страницы «Галерея». Здесь будут отображаться последние, либо (по желанию пользователя) самые популярные загруженные данные — «превью» изображение и небольшое описание, чтобы понять, стоят ли данные внимания. Слева расположено функциональное меню, позволяющее выполнять некоторые действия на данной странице.

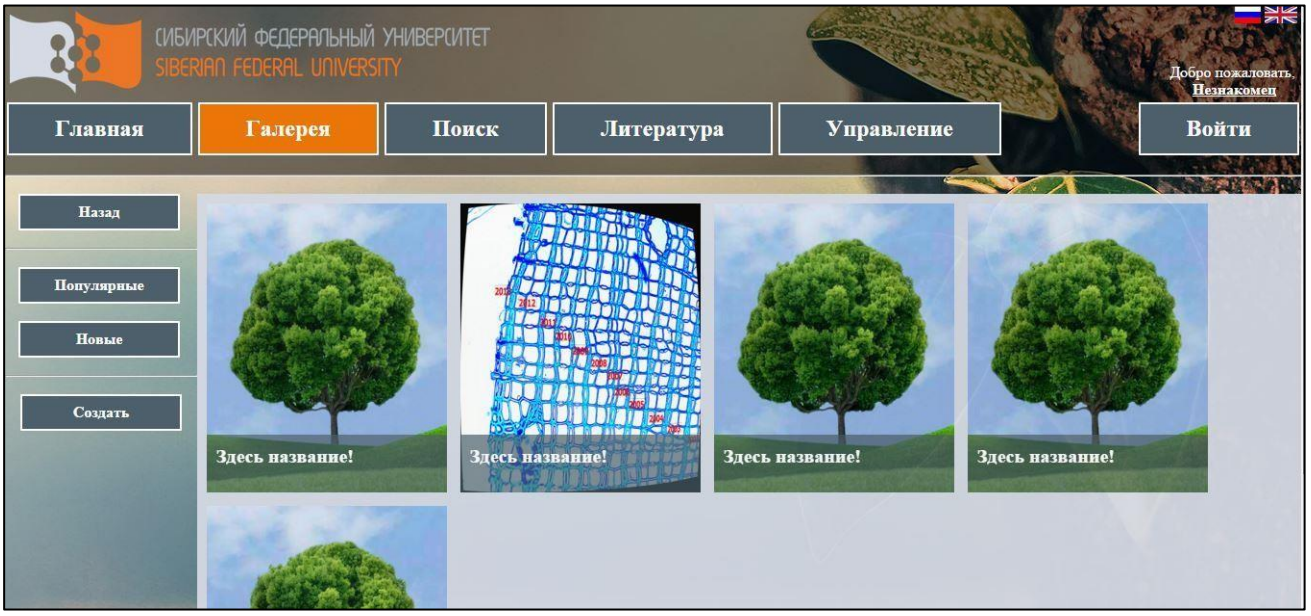


Рисунок 15 — Страница «Галерея»

На рисунке 16 представлено изображение страницы «Профиль». Здесь будет отображаться основная информация о пользователе, некоторую из

которых он сможет скрывать (номер телефона и т.п.). Также здесь можно будет просмотреть, какие материалы пользователь загрузил и какие скачал. Слева расположено функциональное меню, с помощью которого пользователь сможет отредактировать персональную информацию или сменить пароль для входа на сайт.

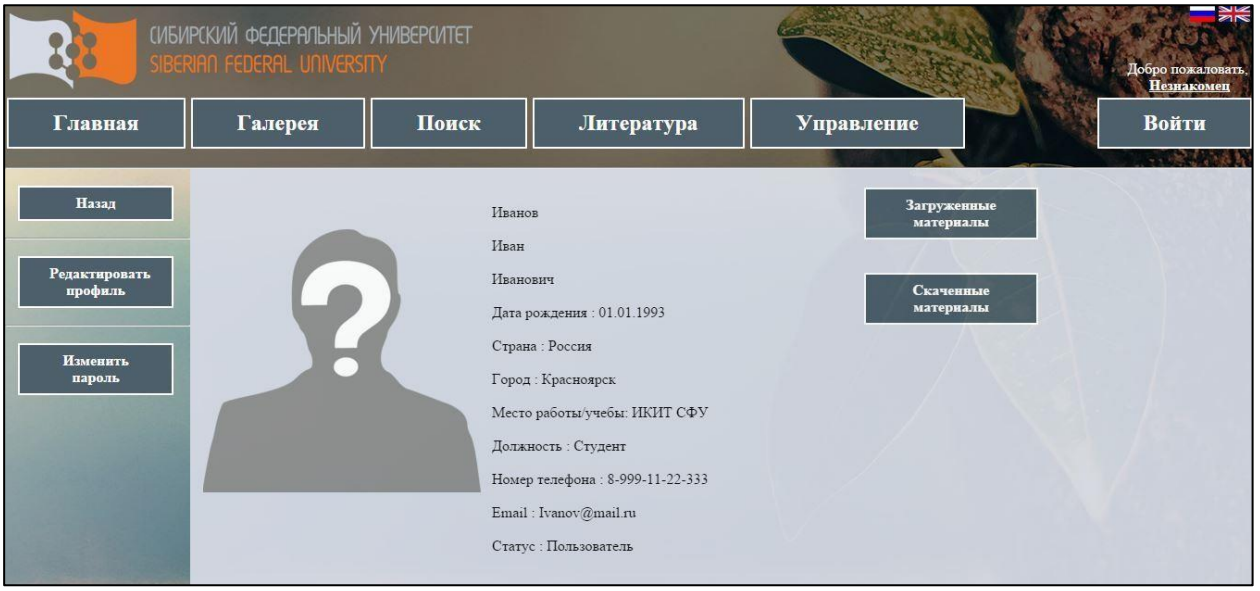


Рисунок 16 — Страница «Профиль»

На рисунке 17 представлено изображение страницы «Поиск». Здесь пользователь может получить список данных, отфильтрованных по предварительно внесенным пользователем критериям.

На изображении представлен прототип будущего поиска.

При дальнейшей разработке он будет основательно переделан, будет добавлена возможность множественного выбора. Будет внедрен виджет, позволяющий наглядно продемонстрировать месторасположения деревьев, а также осуществить ввод параметров поиска по месторасположению одним кликом компьютерной мыши без ручного ввода точных координат.

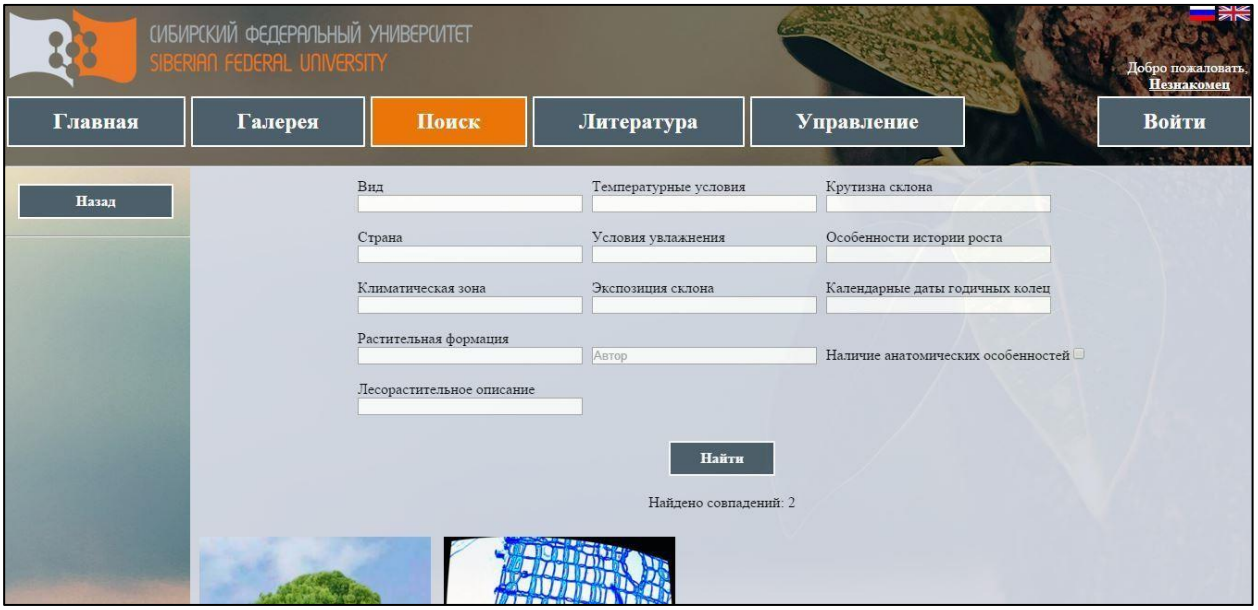


Рисунок 17 — Страница «Поиск»

На рисунке 18 представлено изображение страницы «Управление атрибутами». На этой странице администратор может добавлять новые атрибуты для данных, редактировать их и удалять. Также здесь происходит редактирование справочников атрибутов.

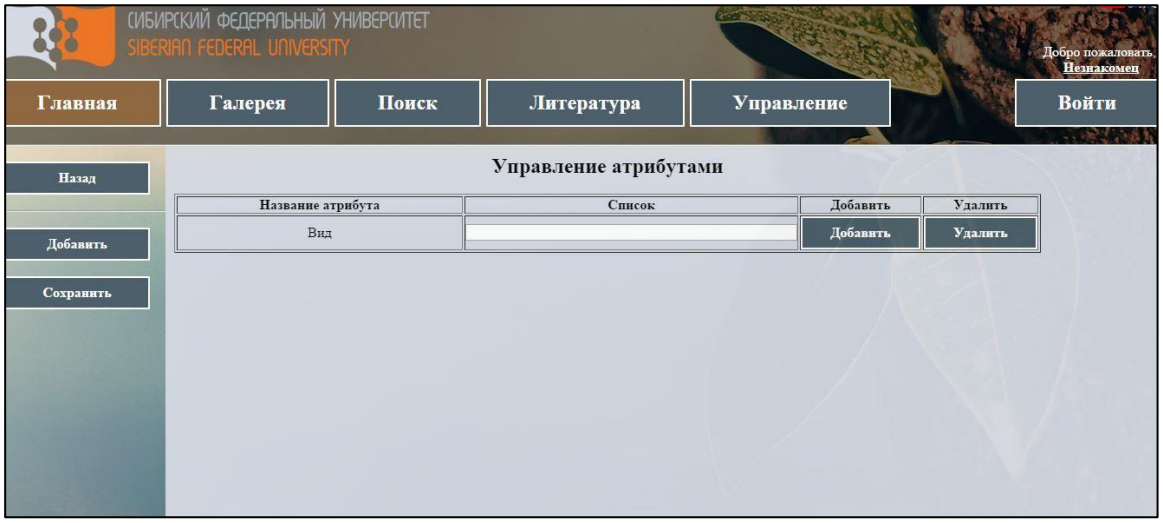


Рисунок 18 — Страница «Управление атрибутами»

На рисунке 19 представлено изображение страницы «Создание данных». На этой странице пользователь может загрузить свои измерения в общую базу.

Прежде чем загрузить данные, он должен заполнить все поля атрибутов и принять лицензионное соглашение.

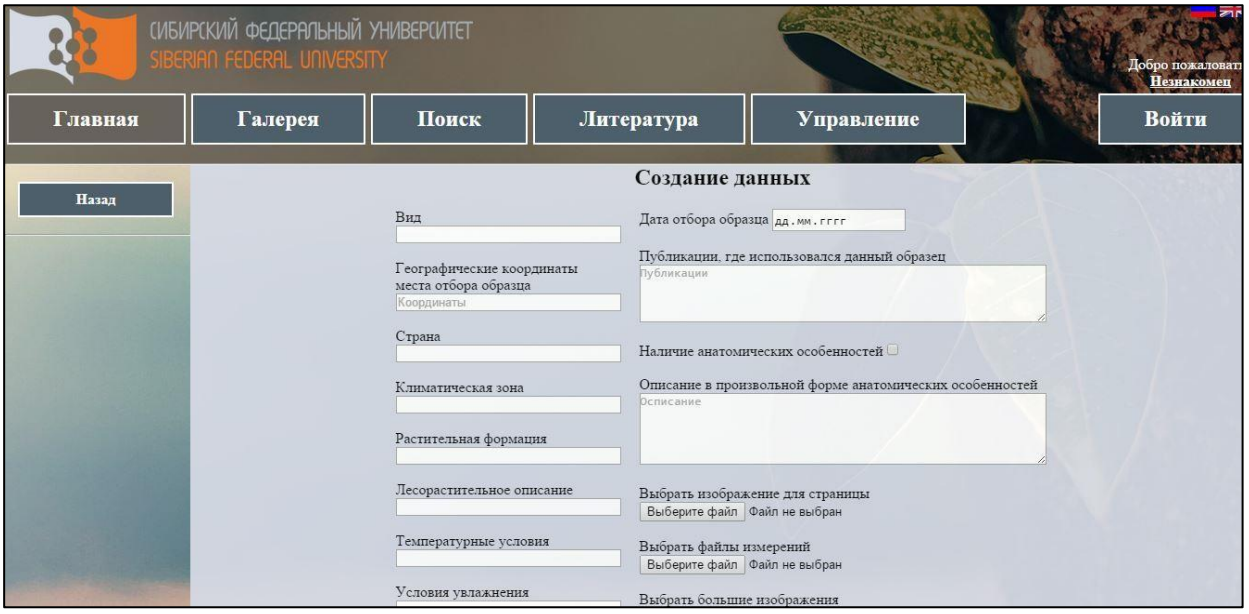


Рисунок 19 — Страница «Создание данных»

После завершения верстки, данный проект был представлен заказчику и был одобрен, как начальный прототип, который впоследствии будет дорабатываться и пополняться новыми функциями.

В дальнейшем произошел перенос верстки на язык scala посредством библиотек ScalaTags и были созданы шаблоны страниц.

3.4 Обмен данными

Данный раздел описывает реализацию обмена данными между БД системы и пользователем посредством ИС.

Прежде всего необходимо определить параметры БД системы в файле *ebean.properties*, такие как имя пользователя БД, пароль. После этого можно подключить БД системы уже к самому проекту, чтобы выполнять SQL-запросы или использовать как-то иначе, но уже без открытия СУБД PostgreSQL. Это особенно удобно, когда требуется регулярно смотреть меняющееся содержимое таблиц.

На рисунке 20 изображена структура подключенной к проекту БД системы.

Рисунок 20 — Структура подключенной БД разрабатываемой системы

В данном случае на изображении уже представлена БД системы, содержащая таблицы с записями в них. Но, чтобы таблицы появились в БД и для сравнительно небольшого объема кода обращений к ним, необходимо сначала реализовать эти методы, используя инструменты ORM-фреймворка Ebean.

Создается scala-класс, который благодаря использованию аннотации java.persistence @Entity представляет из себя сущность. Поля этого класса с помощью аннотаций представляют из себя атрибуты сущности. Также благодаря им задается отношение между таблицами, такие как «многие ко многим», «один ко многим», «многие к одному» или «один к одному» .

Пример такого класса представлен на рисунке 21.

```
@Entity
@Table(name="users")
class User extends Model{
  @Id
  var id: Int = _

  @Column(length = 255)
  var name: String = _

  @Column(length = 255, unique = true)
  var email: String = _

  @Column(length = 127)
  var password: String = _

  @ManyToOne(targetEntity = classOf[Role])
  var role: Option[Role] = _
}
```

Рисунок 21 — Листинг класса User

Можно обратить внимание, что у атрибута *role* присутствует аннотация @ManyToOne (связь многие к одному), что означает, что он представляет из себя *id* одного из экземпляров сущности *Role*.

Теперь на основе этого класса будут созданы SQL-запросы, позволяющие создать и удалить таблицу users. Они будут храниться соответственно в файлах *pg-create-all.sql* и *pg-drop-all.sql*, откуда можно их выполнить.

Пример такого запроса представлен на рисунке 22.

```
create table users (
  id                          serial not null,
  name                        varchar(255),
  email                       varchar(255),
  password                    varchar(127),
  role_id                     integer,
  constraint uq_users_email unique (email),
  constraint pk_users primary key (id)
);
```

Рисунок 22 — Листинг запроса создания таблицы users

После того, как нужные таблицы созданы, можно приступить к реализации обращения к ним.

Простейший пример такого обращения представлен на рисунке 23. На нем представлены функции поиска экземпляра класса *User* по атрибутам *id* и *email*.

```
def userByEmail(email: String): Option[User] = Option(User.email.equalTo(email).findUnique())
def userById(id: Int): Option[User] = Option(User.id.equalTo(id).findUnique())
```

Рисунок 23 — Листинг функций поиска экземпляра класса User

Если поиск осуществлен успешно и экземпляр найден, то к его атрибутам можно делать обращения с целью присваивания, сравнения и т.п.

Примером использования этих функций можно считать авторизацию.

Когда пользователь пытается войти на сайт, функция *userByEmail* проверяет, существует ли вообще такой пользователь в БД системы. Если существует, то дальше идет следующая проверка — сравнение введенного пользователем пароля и пароля, существующего в БД системы.

Далее рассматривается еще одна функция — добавление в таблицу новых записей.

На рисунке 24 представлен подобный пример.

```
def addUser(user: User): Unit = {
  require(user.id == 0, "user already added")
  Db.server.insert(user)
}
```

Рисунок 24 — Листинг функции добавления нового пользователя

На рисунке 25 рассматривается функция изменения данных — смена пароля пользователя, где так же, как и при авторизации, проверяются введенные пароли.

Если пользователем была совершена ошибка при вводе паролей, то система обнаружит ее и выдаст сообщение об ошибке.

Если же все пароли были введены верно, то система преобразует новый пароль методами криптографического шифрования и обновит его в таблице БД системы.

```
def updatePassword(userId: Int, oldPassword: String, newPassword: String)(implicit lang: Lang): UpdateResult = {
  val transaction: Transaction = Db.server.beginTransaction()
  val result = userById(userId) match {
    case Some(user) =>
      if (user.password == oldPassword) {
        user.password = newPassword
        Db.server.update(user)
        UpdateSuccess
      }
      else UpdateFailure(lang.login.incorrectPasswordsForChange)
    case _ => throw new RuntimeException("NO SUCH USER: " + userId)
  }
  transaction.commit()

  result
}
```

Рисунок 25 — Листинг функции смены пароля пользователя

Таким образом происходит взаимодействие пользователя с БД системы посредством ORM-фреймворка Ebean.

3.5 Модуль авторизации

К данному модулю относятся части ИС, связанные с регистрацией, авторизацией, сменой пароля и его восстановлением.

Для пользователя регистрация представляет из себя ввод необходимой достоверной информации в несколько полей. После этого генерируется ссылка, содержащая уникальный токен — длинная строка символов и отправляется на указанную при регистрации почту. Когда пользователь перейдет по ней, регистрация завершится и его данные добавятся в БД системы, причем пароль предварительно перед этим будет зашифрован, используя пару криптографических методов шифрования.

Авторизация — процесс идентификации пользователя и его аутентификации. После того, как пользователь, решив войти в систему, введет свой логин и пароль, система проверит наличие такого пользователя в БД системы, а также сверит введенный пароль и тот, что в БД системы. Если проверка пройдена успешно, то генерируется уникальный токен, частично состоящий из символов хешированного криптографическим методом логина пользователя. Данные пользователя сохраняются в кэше браузера.

При смене пароля просто проверяется правильность ввода действующего пароля и, если все совпадает, в БД системы сохраняется новый зашифрованный пароль, а информация в кэше обновляется.

При восстановлении пароля генерируется ссылка, содержащая уникальный токен, и отправляется на почту пользователя, указанную при регистрации. После перехода по ней дается возможность сменить пароль. Важно отметить, что эта ссылка будет действительна ограниченное время. Это реализовано благодаря созданию таблицы в БД системы, сохраняющей все попытки пользователей восстановить пароль, а также время, когда это произошло.

3.6 Модуль загрузки данных

Основной модуль ИС, представляющий из себя инструмент, позволяющий пользователю загружать свои данные на сервер для хранения или дальнейшего использования. Несмотря на то, что этот модуль является одним из основных, особо сложных процессов здесь не происходит, если считать процесс конвертирования данных из файла формата Excel в формат JSON отдельным модулем.

Вводимые пользователем данные сохраняются в БД системы согласно их типам, файл с измерениями проходит конвертацию и сохраняется в формате JSON, изображение прикрепляется из личного хранилища пользователя.

3.7 Модуль конвертации

Модуль предназначен для преобразования полученного файла измерений в формат JSON для успешного сохранения в БД системы, а также для его восстановления в формат Excel.

В дальнейшем обращение к данным этого файла будет значительно проще и быстрее, нежели если бы файл оставили как есть. Помимо повышения эффективности появляется возможность проверить файл на соблюдение

стандарта, тем самым не позволяя пользователям публиковать в открытом доступе необработанный материал.

Для работы с файлами формате Excel используется библиотека Apache POI. Apache POI — это библиотека, написанная на языке Java, для чтения и записи документов Microsoft Office, таких как Excel, PowerPoint и Word. С помощью библиотеки Apache POI возможно программным способом создавать новые документы или изменять существующие, индексировать текст, обрабатывать вложенные (*embedded*) объекты (документы, картинки и т.д.) и много чего другого [14].

Алгоритм конвертации данных в формат JSON состоит из нескольких циклов, внутри которых производится сравнение по заданным критериям (так как изначально известна структура файла благодаря установке общего стандарта).

Переходя с поля на поле, алгоритм сохраняет полученные значения в экземпляры классов, отвечающих за те или иные поля.

Пример подобного класса представлен на рисунке 26.

Экземпляром данного класса будет являться одно годичное кольцо, со всеми измерениями. Данный класс имеет атрибуты *year* (календарный год), *thickness* (толщина годичного кольца, мкм), *files* (список радиальных файлов).

С помощью аннотации @JsonProperty задается сокращение, которое будет сопровождать конкретное значение в строке формата JSON. Сделано это в целях экономии памяти и эффективности в плане скорости обращения.

Остальные классы являются подобными этому, отличие только в других атрибутах, поэтому подробного рассмотрения не требуют.

В итоге получается один большой список, в который, помимо простых атрибутов, входит вложенный список другого класса, у которого тоже есть простые атрибуты и вложенный список. И так продолжается, пока не дойдет до последнего класса, у которого нет вложенного списка, только присутствуют простые атрибуты.

```java
/**
 * Данные одного годичного кольца
 */
public class YearData {
    /**
     * Календарный год
     */
    @JsonProperty("y")
    public final int year;

    /**
     * Толщина годичного кольца
     */
    @JsonProperty("th")
    public final double thickness;

    /**
     * Список радиальных файлов
     */
    @JsonProperty("fs")
    public final List<FileData> files;

    public YearData(final int year, final double thickness, List<FileData> files) {
        this.year = year;
        this.thickness = thickness;
        this.files = files;
    }
}
```

Рисунок 26 — Листинг класса YearData

После получения всех данных, полученный массив сохраняется в соответствующее поле в таблице measurements в формате JSON.

Конвертация в файл формата Excel из строки формата JSON происходит в обратном порядке. Так же, как и при конвертации из файла формата Excel, здесь присутствует несколько циклов, в результате прохождения которых значения из строки формата JSON записываются в поля файла формата Excel.

3.8 Смена языка

В требованиях к ИС есть пункт о том, что система должна поддерживать два языка — русский и английский. Эта задача выполнена с помощью создания нескольких классов, хранящих в себе все возможные строки сервиса, записанных на обоих языках.

Пример такого класса представлен на рисунке 27.

```
class LoginStrings(locale: Locale) extends Translable(locale) {
  val authorization: String = En("Authorization") ~ Ru("Авторизация")
  val email: String = En("Email") ~ Ru("Email")
  val password: String = En("Password") ~ Ru("Пароль")
  val submit: String = En("Submit") ~ Ru("Войти")
  val incorrectCredentials: String = En("Invalid email or password") ~ Ru("Неправильно введён логин и
  val passwordRecovery: String = En("Password recovery") ~ Ru("Восстановление пароля")
  val recovery: String = En("Recovery") ~ Ru("Восстановить")
  val yourEmail: String = En("Your email") ~ Ru("Ваш email")
  val yourPassword: String = En("Your password") ~ Ru("Ваш пароль")

  val passwordChange: String = En("Change password") ~ Ru("Смена пароля")
  val oldPassword: String = En("Your old password") ~ Ru("Ваш прежний пароль")
  val newPassword: String = En("Your new password") ~ Ru("Ваш новый пароль")
  val repeatNewPassword: String = En("Repeat your new password") ~ Ru("Повторите ваш новый пароль")
  val change: String = En("Change") ~ Ru("Сменить")
  val incorrectPasswordsForChange: String = En("Invalid old password or new password doesn't match")

  val incorrectEmail: String = En("Invalid email") ~ Ru("Неправильно введён логин")

}
```

Рисунок 27 — Листинг класса LoginStrings

Итогом этой главы является спроектированная база данных цифровой микроанатомии годичных колец, а также спроектированная и разработанная ИС.

3.9 Вывод по главе 3

Итогом этой главы является разработанная БД системы и ИС. На данный момент реализованы не все функции, указанные в требованиях, но сама ИС разработана и остается только создавать новые модули и внедрять в нее.

Реализована основная функция — загрузка данных. Зарегистрированный пользователь может загрузить файл измерений с указанием обязательных атрибутов, а система автоматически сформирует из этого файла строку данных в формате JSON и сохранит ее вместе с атрибутами в БД системы.

Впоследствии, после реализации функции отображения данных, эта информация будет запрашиваться из БД системы, и пользователь сможет увидеть ее на экране своего монитора.

ЗАКЛЮЧЕНИЕ

В результате бакалаврской работы выполнены поставленные задачи. Выявлены и проанализированы требования к информационной системе, построены UML-диаграммы. Спроектирована и разработана база данных цифровой микроанатомии годичных колец. Спроектирована и разработана информационная система.

В процессе разработки информационной системы созданы модули авторизации, загрузки данных, конвертации.

В будущем планируется доработать информационную систему, чтобы она соответствовала всем требованиям и имела все необходимые функции, в частности функцию поиска.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Матвеев, С. М. Дендрохронология : учебное пособие. — 2-е изд. / С.М. Матвеев, Д.Е. Румянцев ; М-во образования и науки РФ, ФГБОУ ВПО «ВГЛТА». — Воронеж, 2013. — 140с.

2. Карточка проекта, поддержанного Российским научным фондом [Электронный ресурс] // grant.rscf.ru: Российский научный фонд. — Режим доступа : http://grant.rscf.ru/prjcard?rid=15-14-30011

3. The International Tree-Ring Data Bank [Электронный ресурс] // utc.edu: The University of Tennessee. — Режим доступа : http://web.utk.edu/~grissino/itrdb.htm

4. Фаулер, М. UML. Основы. Краткое руководство по унифицированному языку моделирования : книга. — 3-е изд. / Мартин Фаулер. — СПб: Символ-Плюс, 2004. — 192 с.

5. Сомасегар, С. Руководство Microsoft по проектированию архитектуры приложений : практическое руководство. — 2-е изд. / С. Сомасегар, С. Гатри, Д. Хилл ; Microsoft. — 2009. — 529с.

6. Подробнее о технологии Java [Электронный ресурс] // java.com: Oracle. — Режим доступа : https://www.java.com/ru/about/

7. Object-Oriented Meets Functional [Электронный ресурс] // scala-lang.org : Scala. — Режим доступа : http://www.scala-lang.org/

8. The High Velocity Web Framework For Java and Scala [Электронный ресурс] // playframework.com : Typesafe, Ink. — Режим доступа : https://www.playframework.com/

9. What is spray? [Электронный ресурс] // spray.io: Typesafe, Ink. — Режим доступа : http://spray.io/introduction/what-is-spray/

10. ScalaTags [Электронный ресурс] // github.com. — Режим доступа : http://lihaoyi.github.io/scalatags/

11. Documentation / Introduction [Электронный ресурс] // github.com. — Режим доступа : http://ebean-orm.github.io/docs/introduction

12. PostgreSQL [Электронный ресурс] // postgresql.org : The PostgreSQL Global Development Group. — Режим доступа : http://www.postgresql.org/

13. HTML и CSS [Электронный ресурс] : Сайт-учебник по HTML и CSS. — Режим доступа : http://htmlbook.ru/

14. Новая версия Apache POI [Электронный ресурс] : habrahabr.ru : Тематические медиа. — Режим доступа : https://habrahabr.ru/post/140898/

15. СТО 4.2-07-2014 : Стандарт организации. — Общие требования к построению, изложению и оформлению документов учебной деятельности. — Система управления СФУ. — Красноярк, 2014. — 60с